ROOT: superbuilds

PAVLO SVIRIN, 2023-12-06

Personal information

- PhD: National Technical University of Ukraine (2014), Computer Science
- Academic work experience:
  - CERN (2014-2017): project associate at ALICE experiment
  - Brookhaven National Laboratory (2017-2019)
  - CERN (2019-2021): project associate at ATLAS experiment
  - Barcelona Supercomputing Center (2021-2023)
- Speaks Ukrainian, English, Spanish, Chinese, Russian. Some knowledge about Sanksrit, Middle Egyptian, Crimean Tatar.

ROOT

- ROOT is a framework for data processing developed at CERN
- Used in high-energy physics and astrophysics
- Provides lots of features for:
  - data processing
  - data saving and data access
  - publish results
  - using interactive sessions using Cling C++ or building customs applications
- Website: https://root.cern/

ROOT: simplification of compilation

- ROOT needs lots of time to compile and user not all of the modules
  - Around 130 internal modules with inter-dependencies
- Practical use case: instead of downloading more then 1GB of full ROOT sources or pre configured ROOT binaries, you can decide to start with minimal set ~50 Mb and expand with any customization you want.

ROOT: simplification of compilation

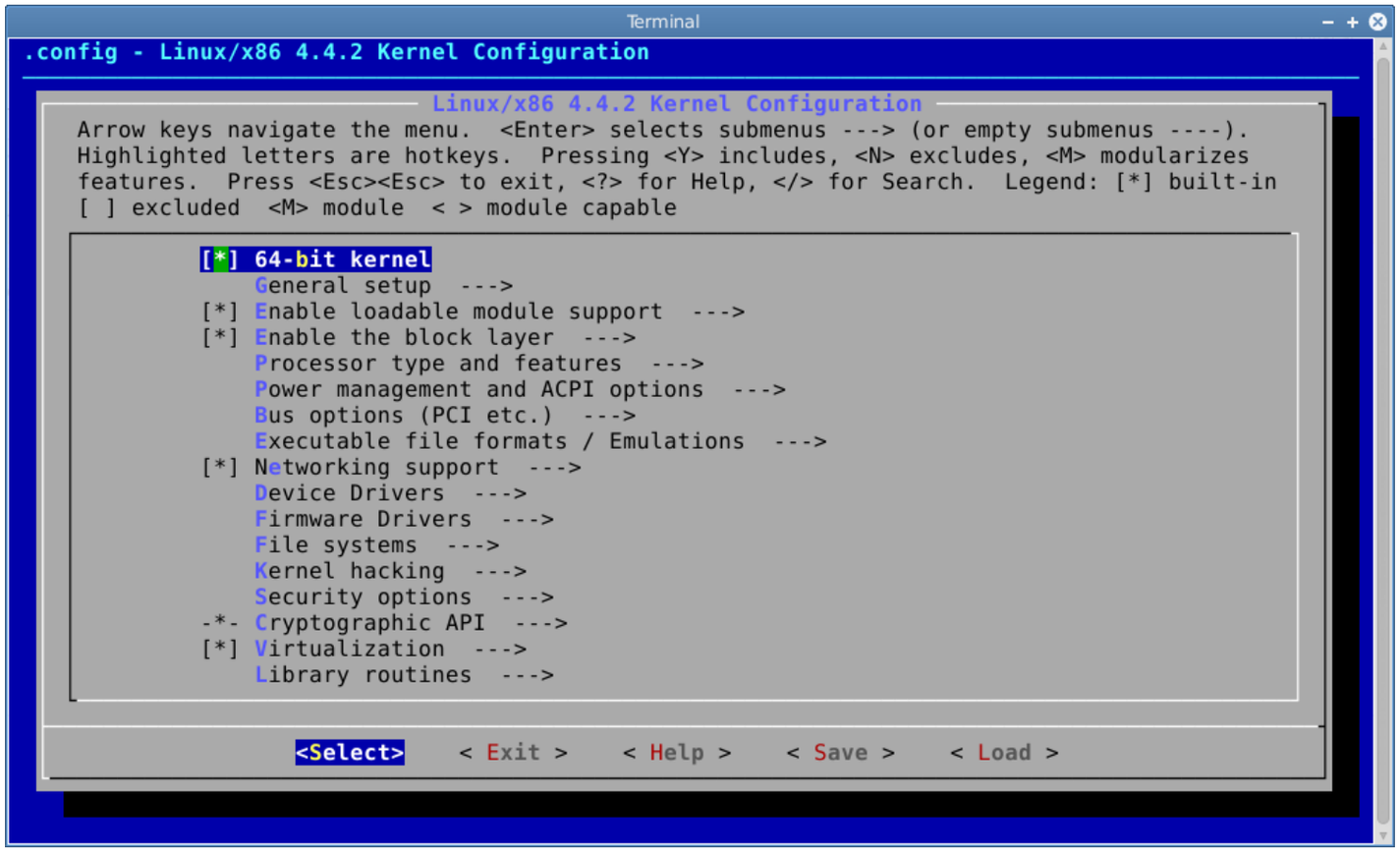
- The idea is to specify which components have to be compiled during configuration time
- Auto-detection of dependencies among the modules
  - done by parsing of CMakeLists files in search of ROOT_STANDARD_LIBRARY definitions and their dependencies
  - Dependency tracking can be implemented using simple graph database like https://github.com/dpapathanasiou/simple-graph
- Absolutely minimal set of module to be compiled to run ROOT:
  - Core, IO, CLING interpreter, MathCore
  - other modules compiled if specified

ROOT: menu-based compilation

- Cmake call will look like the following:

```
cmake ../root-6.28.06/ -Dxrootd=0 -Dssl=0 -Dtmva=0 -Dwebgui=0 -Dxproofd=0 -Dgraf=0 -Dexecutables=1
-Dnet=1 -Ddb=1 -Dmath=1 -Dbindings=1 -Dhtml=0 -Dgui=0 -DCMAKE_INSTALL_PREFIX=/mnt/sdb1/opt/
root-modules -Dxml=0 -Dhttp=0 -Dtree=0 -Dproof=0 -Druntime_cxxmodules=1
```

- The idea is to develop a similar to Linux's menuconfig TUI tool which will automatically enable necessary dependencies from selections



Distributed modulemap files

- Modulemap in ROOT is a file which defines available components in the installation directory, their headers and shared libraries
- Currently include/module.modulemap a file of several hundreds lines
- We managed to split it into multiple files:
  - each file defines one component
  - main modulemap file just includes all of these files
- Benefits:
  - easy to add new components
  - easy to identify which components are already installed

ROOT: partial builds

- Goal:
  - to allow to skip compilation of the components which are already built and installed to target directory
  - to easily add new components to distributed modulemap infrastructure
  - in case of admin-only rights to write into ROOT’s installation directory: to install new components together with their modulemap files to different directory and then on ROOT’s start combine all of the necessary modulemaps into one

ROOT: problem example

- “Cannot build ROOT if another ROOT at /usr/local” https://github.com/root-project/root/issues/7128
- This is relevant for
  - sanity: separating the build and its artifacts from other, pre-existing ROOT artifacts allows us to be certain we build what we think we build and we test the build and not a combination of the build and whatever other ROOT there is;
  - Target audience: people building ROOT on shared resources for which they don't have admin powers
  - Expected behavior: ROOT builds should be the same whether or not there are other ROOT installs / binaries no the system

Conclusions and future work

- Completed:
  - builds with specified components for basic components
    - complex components like ones for plotting require more sophisticated approach
  - build with minimal number of components was created and successfully tested
  - a tool which scans ROOT libraries for dependencies was implemented and tested
- In progress:
  - incremental builds, which allow to skip already installed components
  - tracking dependencies among complex components